CLAIMS

What is claimed is:

1. In a threaded computing environment having a plurality of contexts, each context capable of containing a queue, context settings, a context dictionary, and objects, a method for allocating the access of threads to a user interface context, the method comprising:

receiving a request to access the user interface context from a first thread;

determining whether the user interface context is presently being accessed by a second thread, and:

if the user interface context is presently being accessed by a second thread, denying the request to access the user interface context received from the first thread; and

if the user interface context is not presently being accessed by a second thread, allowing the request to access the user interface context received from the first thread.

2. The method for allocating the access of threads to user interface context of claim 1, the method further comprising:

maintaining a context record associated with each thread that identifies the contexts accessed by the thread, the most recent entry in the context record indicating the context presently being accessed by the thread;

when a thread accesses an object in the user interface context, checking the most recent entry in the context record associated with the thread;

determining whether the most recent entry in the context record matches the context of the object being accessed; and

if the most recent entry in the context record does not match the context of the object being accessed, raising an exception.

3. The method for allocating the access of threads to a user interface context of claim1, the method further comprising:

maintaining thread settings associated with threads;

maintaining context settings in the user interface context; and

applying the context settings of the user interface context in place of the thread settings of any thread accessing the user interface context.

4. The method for allocating the access of threads to a user interface context of claim 3, the method further comprising restoring the thread settings when a thread departs the user interface context.

5. In a threaded computing environment having a plurality of contexts, context being capable of containing a queue, context settings, a context dictionary, and objects, a method for controlling the processing of jobs in a context of the computing environment, the method comprising:

receiving jobs from a thread, a job received from a thread comprising:

a process to be executed;

a priority, the priority being one of a plurality of predefined priority levels ranging from a highest priority to a lowest priority; and

a receipt time;

maintaining received jobs in a first queue within the context, the first queue comprising a record of received jobs, a record of received job comprising:

the process to be executed for the job;

the priority of the job; and

the receipt time of the job;

processing jobs maintained in the first queue, processing jobs maintained in the queue comprising:

processing jobs in priority order, from the highest priority to the lowest priority;

for jobs of equal priority, processing jobs in receipt time order, from the earliest receipt time to the latest receipt time; and

performing the process to be executed for a job in the queue when the job is the next to be processed, the jobs in order; and

removing a job from the first queue after the job has been processed.

6. The method for controlling the processing of jobs of claim 5, wherein:

a job received from a thread further comprises:

a trigger, the trigger defining an event which will require an action to be taken with the job within the first queue if the event occurs; and

an action to be taken with the job within the first queue if the trigger occurs;

a record of a received job further comprises:

the trigger for the job; and

the action to be taken with the job if the trigger occurs; and

processing jobs maintained in the print queue further comprises:

for each job in the queue, if the trigger occurs, performing the action to be taken with the job when a trigger occurs.

7. The method for controlling the processing of jobs of claim 6, wherein the action to be taken with a job if a trigger occurs is selected from:

aborting the job by removing the job from the first queue; and

changing the priority of a job within the first queue without changing the receipt time of the job.

8. The method for controlling the processing of jobs of claim 6, wherein the trigger comprises the expiration of a predetermined amount of time after the job was placed in the queue.

9. The method for controlling the processing of jobs of claim 6, wherein the trigger comprises the completion of other processing.

10. The method for controlling the processing of jobs of claim 6, wherein the trigger comprises the receipt of a notification from the operating system.

11. The method for controlling the processing of jobs of claim 5, further comprising:

processing jobs in the queue of each context with at least one dispatcher, each context having a dispatcher that processes the jobs in the queue of the context, processing each job comprising assigning the job to a thread.

12. The method for processing jobs from a queue within the queue of at least one context within a computing environment of claim 11, further comprising;

pinning at least one context to a specific thread, such that all jobs in the queue of the at last one context are assigned to the specific thread.

13. A computer readable media on which is stored computer readable code to cause a computer to perform a method for allocating the access of threads to a user interface context in a threaded computing environment having a plurality of contexts, each context capable of containing a queue, context settings, a context dictionary, and objects, the method for allocating the access of threads to a user interface context comprising:

receiving a request to access the user interface context from a first thread;

determining whether the user interface context is presently being accessed by a second thread, and:

if the user interface context is presently being accessed by a second thread, denying the request to access the user interface context received from the first thread; and

if the user interface context is not presently being accessed by a second thread, allowing the request to access the user interface context received from the first thread.

14. The computer readable media of claim 13, the method for allocating the access of threads to a user interface further comprising:

maintaining a context record associated with each thread that identifies the contexts accessed by the thread, the most recent entry in the context record indicating the context presently being accessed by the thread;

when a thread accesses an object in the user interface context, checking the most recent entry in the context record associated with the thread;

determining whether the most recent entry in the context record matches the context of the object being accessed; and

if the most recent entry in the context record does not match the context of the object being accessed, raising an exception.

15. The computer readable media of claim 13, the method for allocating the access of threads to a user interface comprising:

maintaining thread settings associated with threads;

maintaining context settings in the user interface context; and

applying the context settings of the user interface context in place of the thread settings of any thread accessing the user interface context.

16. The computer readable media of claim 15, the method for allocating the access of threads to a user interface further comprising restoring the thread settings when a thread departs the user interface context.

17. A computer readable media on which is stored computer readable code to cause a computer to perform a method for controlling the processing of jobs in a context of a computing environment having a plurality of contexts, contexts being capable of containing a queue, context settings, a context dictionary, and objects, the method for controlling the processing of jobs in a context comprising:

receiving jobs from a thread, a job received from a thread comprising:

a process to be executed;

a priority, the priority being one of a plurality of predefined priority levels ranging from a highest priority to a lowest priority; and

a receipt time;

maintaining received jobs in a first queue within the context, the first queue comprising a record of received jobs, a record of received job comprising:

the process to be executed for the job;

the priority of the job; and

the receipt time of the job;

processing jobs maintained in the first queue, processing jobs maintained in the queue comprising:

processing jobs in priority order, from the highest priority to the lowest priority;

for jobs of equal priority, processing jobs in receipt time order, from the earliest receipt time to the latest receipt time; and

performing the process to be executed for a job in the queue when the job is the next to be processed, the jobs in order; and

removing a job from the first queue after the job has been processed.

18. The computer readable media of claim 17, the method for controlling the processing of jobs in a context further comprising:

a job received from a thread further comprises:

a trigger, the trigger defining an event which will require an action to be taken with the job within the first queue if the event occurs; and

an action to be taken with the job within the first queue if the trigger occurs;

a record of a received job further comprises:

the trigger for the job; and

the action to be taken with the job if the trigger occurs; and

processing jobs maintained in the print queue further comprises:

for each job in the queue, if the trigger occurs, performing the action to be taken with the job when a trigger occurs.

19. The computer readable media of claim 18, wherein the action to be taken with a job if a trigger occurs is selected from:

aborting the job by removing the job from the first queue; and

changing the priority of a job within the first queue without changing the receipt time of the job.

20. The computer readable media of claim 18, wherein the trigger comprises the expiration of a predetermined amount of time after the job was placed in the queue.

21. The computer readable media of claim 18, wherein the trigger comprises the completion of other processing.

22. The computer readable media of claim 18, wherein the trigger comprises the receipt of a notification from the operating system.

23. The computer readable media of claim 17, the method for controlling the processing of jobs in a context further comprises:

processing jobs in the queue of each context with at least one dispatcher, each context having a dispatcher that processes the jobs in the queue of the context, processing each job comprising assigning the job to a thread.

24. The computer readable media of claim 23, the method for controlling the processing of jobs in a context further comprising:

pinning at least one context to a specific thread, such that all jobs in the queue of the at last one context are assigned to the specific thread.